1a)
You can determine the type of device that the page is being displayed on using media queries. All devices have different screen sizes, specifically width. With media queries you can detect the maximum or minimum size of the screen the page is being displayed on and as a result you can determine what type of device the page is on (phone, laptop, desktop monitor).
As a developer, you would care because the way that the front end is displayed on a giant monitor would not be the same on a small phone screen. Being able to adapt and change the layout of the page depending on the type of device is very important for allowing users to easily navigate a website.
If you were to implement UI elements with extra animations and feature such as a hover, it would not work on a mobile device because there is no mouse to hover with. A developer would have to take in account what would be feasible on a phone touch screen or a laptop/desktop

1b)
A package-lock.json file records the exact version of a package/module/dependency that you install. The file is used so that you can replicate the exact environment later on a different machine or even a different(new) application. The file is not required, but when working on a project with multiple developers, keeping a consistent version runs into less issues with developing an app (outdated versions vs latest versions of a package).

1c)
Npm is an acronym for Node Package Manager. It is used to be able to install node applications, along with other packages and modules and dependencies for an API
Npm is also used to run a server for your app (npm server.js) There are many useful packages and modules within npm that enhance a developers application for both the user experience and the developer experience. For example, nodemon is a package that automatically saves and rebuilds the application on the server with the newest changes so you do not have to restart the server after every change.

1d)

From the front end, the user could submit a form, press a button, etc. They may enter certain inputs to declare which type of request they want to make (GET, POST, PUT, DELETE) and provide additional information to explain the kind of information they want associated with the method. After a certain action is performed, the front end will make a request (in examples from class via ajax) to an external entity given a specific url. Using the API implemented with the node server, the provided request header will be processed by the app and will make the http request to the external entity. Afterwards, the node server will return the response information and data back to the client that made the request. After receiving the information, the page can use it and display it on the front end.

4

```
var net = require('net');
var socket = require('socket.io')(net); //needed this line of code

var sockets==[];

var s = net.Server(function(socket) {
  sockets.push(socket);

  socket.on('data', function(d) {
    for(var i=0; i<sockets.length;i++) {
      if (sockets[i]==socket) continue;
      sockets[i].write(d);
    }
  });
  socket.on('end', function() {
    var i=sockets.indexOf(socket);
    sockets.splice(i,1);
  });
});

s.listen(8080);
```

The code is an application that requires a network to make transactions between users and the server. On the server, the app puts an initial socket into a list. When a function ‘data’ is called, the server writes data ‘d’ to a place in the list of sockets where it is not the same as the initial socket. When an event ends the server, deletes the socket on the list of sockets where it was originally placed. The app is hosted on port :8080

The app requires the net module in order for the user to interact with a server
A list of sockets is initialized. On the server a socket is pushed to the list of sockets. On event name ‘data’ with the input d, we check the entire list of sockets. If a position on the array is the same as the socket from the server connect we keep iterating through the array, if not we write the value d into that position in the list of sockets. On event name ‘end’ a variable i is initialized containing the index of which position the socket is contained in the list of sockets. We remove the socket from the list of sockets at the index. The app is hosted on port :8080